Second-Order Masked Lookup Table Compression Scheme

Annapurna Valiveti, Srinivas Vivek

IIIT Bangalore

annapurna@iiitb.org, srinivas.vivek@iiitb.ac.in

14-17 September, CHES 2020

Introduction

Side-Channel Attacks

- Traditionally, cryptosystems were viewed as black boxes
- Change of view in the crypto research community since mid-90s due to Kocher *et al.*

Side-Channel Attacks

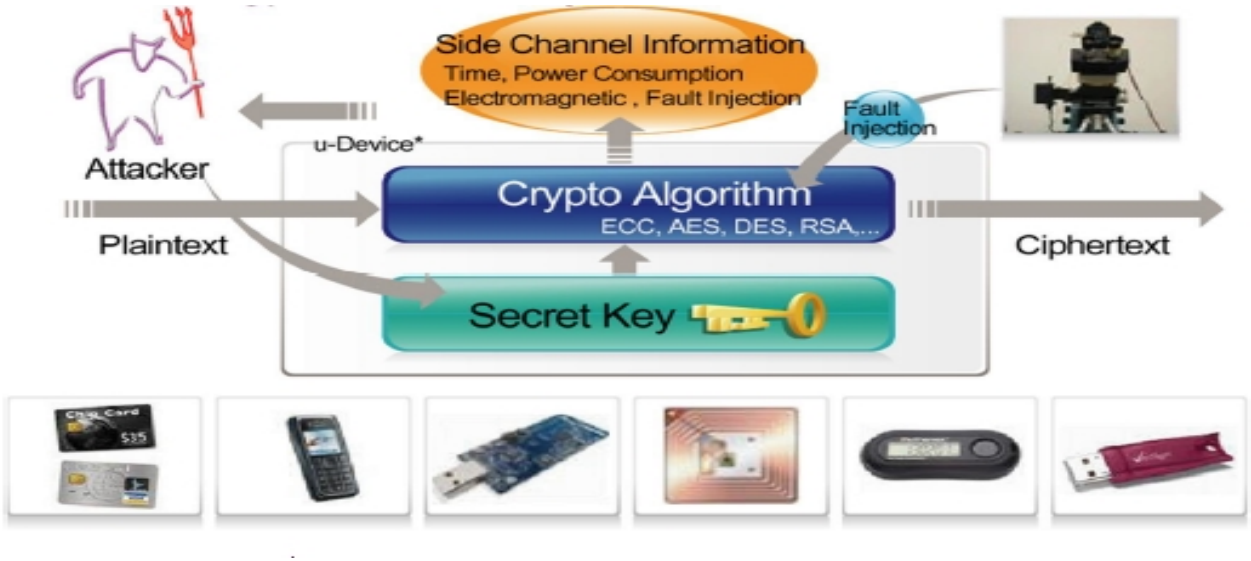


Figure: Side-channel experiment

Side-Channel Attacks

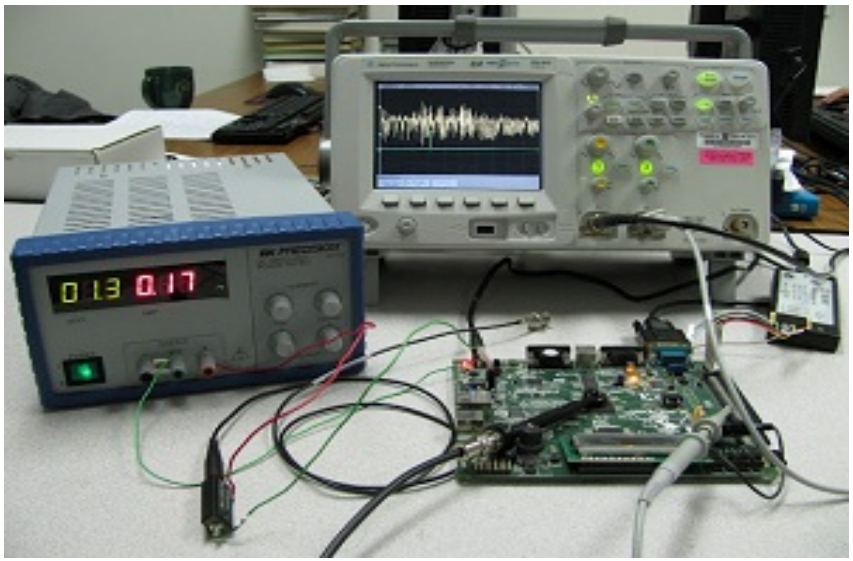


Figure: Power attack setup

Masking Countermeasure

- In this presentation, we only focus on software countermeasures to power analysis attacks
- *Goal* is to minimise the effect of side-channel leakage
- Masking countermeasure against SCA
  - $x = x_1 \oplus \ldots \oplus x_d \oplus x_{d+1}$
  - $d \leftarrow$ masking order

Security Models

- Loosely speaking, SCA complexity is exponential w.r.t. masking order d
- Security offered has been relatively well analysed
- *Probing* [*ISW'03*] & *noisy* leakage model [*CJJR'99, RP'13, DDF'14*]

$$x = x_1 \oplus x_2 \ldots x_d \oplus x_{d+1}$$

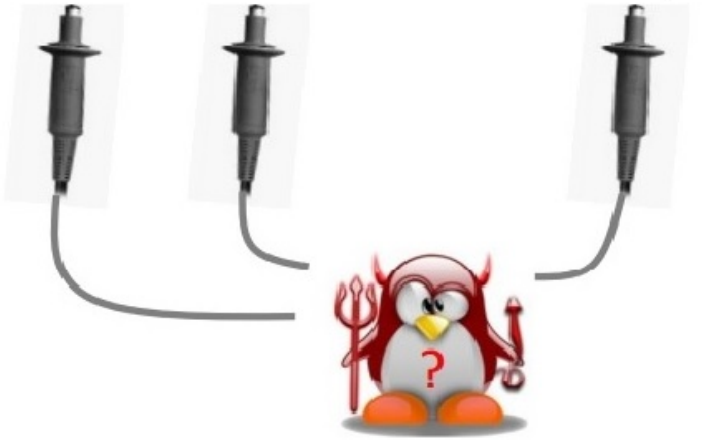


Figure: Adversary observing using at most d probes

Masking of Block Ciphers

Categories of block cipher operations:

- *Linear* functions are straightforward to compute in presence of shares
  - $f(x) = f(x_1) \oplus f(x_2) \oplus \ldots \oplus f(x_{d+1})$
- Main challenge is to securely compute *non-linear* functions
  - For block ciphers, this reduces to securing their S-boxes

Classification of Countermeasures

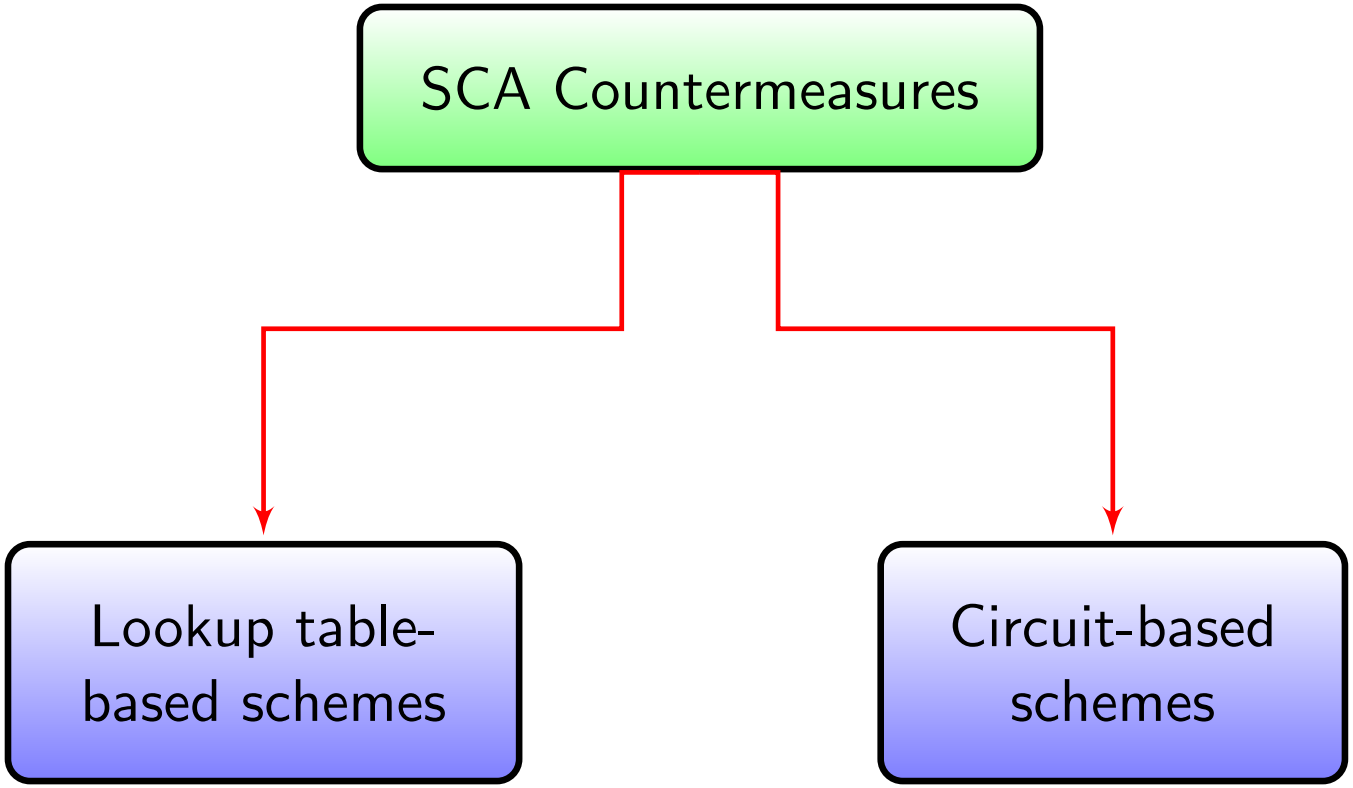


Figure: Classification of countermeasures

First-Order Table-Based Masking

First-order (1-O) lookup table masking. Originally proposed in [*CJJR'99*]

- **Input**:
  - (n, m)-S-box
  - Two input shares x_1, x_2, s.t. $x = x_1 \oplus x_2$
- **Method**:
  - Create a temporary table T in RAM s.t.

$$T(a) = S(x_1 \oplus a) \oplus y_1 \qquad \forall\ a \in \{0,1\}^n$$

  - *Output shares*: y_1, $y_2 = T(x_2)$, s.t. $S(x) = y_1 \oplus y_2$

Table-Based vs. Circuit-Based S-Box Masking

- **AES**: time overhead factor: **2 to 4**, RAM memory = **256 bytes** per S-box function
- RAM Memory can be *expensive* for highly resource-constrained environments
- Alternate approaches exist ([*PR'07*]): $O(1)$ RAM but time overhead factor ≥ 30

Time vs. Memory

Time vs. Memory

Lookup Table Compression Schemes

Lookup Table Compression

- A first-order lookup table compression scheme was proposed in [*RRST'02*]

Lookup Table Compression

- A first-order lookup table compression scheme was proposed in [*RRST'02*]
- An *improved* lookup table compression scheme was by Vadnala [*Vad'17*]

Lookup Table Compression

- A first-order lookup table compression scheme was proposed in [*RRST'02*]
- An *improved* lookup table compression scheme was by Vadnala [*Vad'17*]

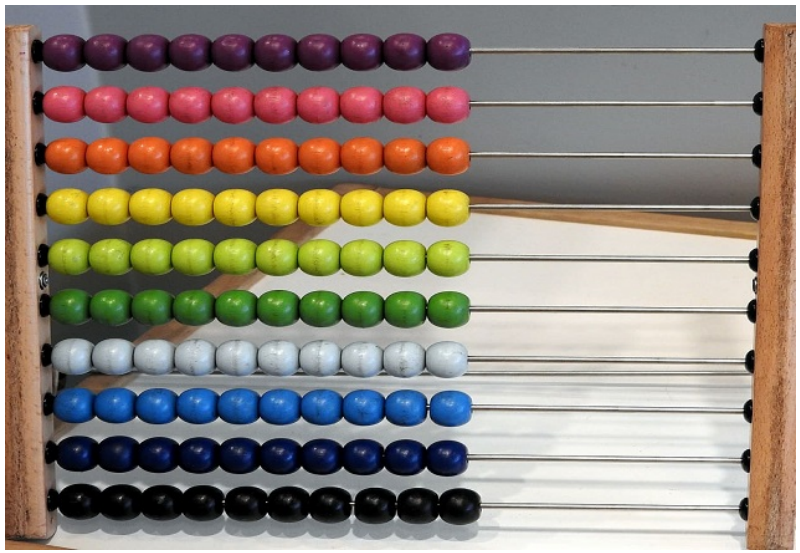


Figure: Pack entries based on higher-order bits

[Vad17] 1-O Table Compression Scheme

- *Partition* the original table T into T_1 and T_2 using compression parameter, ℓ
- $$a = \underbrace{a^{(1)}}_{n-\ell} \,||\, \underbrace{a^{(2)}}_{\ell}$$

[Vad17] 1-O Table Compression Scheme

- *Partition* the original table T into T_1 and T_2 using compression parameter, ℓ
- $$a = \underbrace{a^{(1)}}_{n-\ell} \,||\, \underbrace{a^{(2)}}_{\ell}$$
- Pack 2^{ℓ} distinct entries of T into each row of T_1

[Vad17] 1-O Table Compression Scheme

- *Partition* the original table T into T_1 and T_2 using compression parameter, ℓ
- $$a = \underbrace{a^{(1)}}_{n-\ell} \,||\, \underbrace{a^{(2)}}_{\ell}$$
- Pack 2^ℓ distinct entries of T into each row of T_1
- Unpack one of the entries of T_1 into 2^ℓ rows of T_2

[Vad17] 1-O Table Compression Scheme

- *Partition* the original table T into T_1 and T_2 using compression parameter, ℓ
- $$a = \underbrace{a^{(1)}}_{n-\ell} \,||\, \underbrace{a^{(2)}}_{\ell}$$
- Pack 2^ℓ distinct entries of T into each row of T_1
- Unpack one of the entries of T_1 into 2^ℓ rows of T_2
- There is a set of shared random values across T_1 and T_2

2-O Table Compression Scheme [Vad17]

Base scheme used in [*Vad'17*] is [*RDP'08*]

Three steps of the second-order scheme

1: Create Table $T_1 : \{0,1\}^{n-\ell} \rightarrow \{0,1\}^m$

2-O Table Compression Scheme [Vad17]

Base scheme used in [*Vad'17*] is [*RDP'08*]

Three steps of the second-order scheme

1: Create Table $T_1 : \{0,1\}^{n-\ell} \rightarrow \{0,1\}^m$

$$T_1(b^{(1)} :) = \left(\left(\bigoplus_{i \in \{0,1\}^\ell} S((x_3{}^{(1)} \oplus a^{(1)} \oplus r_i) \mid\mid i) \right) \oplus y_1 \right) \oplus y_2$$

2-O Table Compression Scheme [Vad17]

Base scheme used in [*Vad'17*] is [*RDP'08*]

Three steps of the second-order scheme

1: Create Table $T_1 : \{0,1\}^{n-\ell} \rightarrow \{0,1\}^m$

2: Create Table $T_2 : \{0,1\}^{\ell} \rightarrow \{0,1\}^m$

2-O Table Compression Scheme [Vad17]

Base scheme used in [*Vad'17*] is [*RDP'08*]

Three steps of the second-order scheme

1: Create Table $T_1 : \{0,1\}^{n-\ell} \rightarrow \{0,1\}^m$

2: Create Table $T_2 : \{0,1\}^{\ell} \rightarrow \{0,1\}^m$

$$T_2(b^{(2)}) := T_1(v^{(1)} \oplus r_{(x_3{}^{(2)} \oplus a^{(2)})}) \oplus \bigoplus_{j \in \{0,1\}^{\ell}, j \neq a^{(2)}} S_{(x_3{}^{(2)} \oplus j)}(x^{(1)} \oplus r_{(x_3{}^{(2)} \oplus a^{(2)})} \oplus r_{(x_3{}^{(2)} \oplus j)})$$

2-O Table Compression Scheme [Vad17]

Base scheme used in [*Vad'17*] is [*RDP'08*]

Three steps of the second-order scheme

1: Create Table $T_1 : \{0,1\}^{n-\ell} \rightarrow \{0,1\}^m$

2: Create Table $T_2 : \{0,1\}^{\ell} \rightarrow \{0,1\}^m$

3: Access Table T_2 to compute the third output share

$$y_3 = T_2(v^{(2)})$$

Attack on 2-O Scheme of [Vad'17] by [Viv'17]

- First-order scheme is proven to be secure
- [*Viv'17*] pointed a second-order attack which show that any pair of entries in Table T_2 jointly leak up to $n - \ell$ bits of input

Lemma

Let $\beta_1, \beta_2 \in \{0,1\}^l$. *Then*

$$T_2(\beta_1) \oplus T_2(\beta_2) = S(x^{(1)} \,||\, (\beta_1 \oplus x^{(2)} \oplus v^{(2)})) \oplus S(x^{(1)} \,||\, (\beta_2 \oplus x^{(2)} \oplus v^{(2)}))$$

Our Contribution

Second-Order Lookup Table Compression

- Following are the highlights of our scheme
  - Randomise rows of tables T_1 and T_2

Second-Order Lookup Table Compression

- Following are the highlights of our scheme
  - Randomise rows of tables T_1 and T_2
  - **Randomness complexity**:
    $\left((2^{\ell} \cdot (n-\ell)) + m \cdot (2^{(n-\ell)} + 2^{\ell})\right)$-bits

Second-Order Lookup Table Compression

- Following are the highlights of our scheme
  - Randomise rows of tables T_1 and T_2
  - **Randomness complexity**:
    $\left((2^{\ell} \cdot (n - \ell)) + m \cdot (2^{(n-\ell)} + 2^{\ell})\right)$-bits
  - Use *three-wise* independent PRG [*TS09, IKL+13*] to reduce number of true random values

Second-Order Lookup Table Compression

- Following are the highlights of our scheme
  - Randomise rows of tables T_1 and T_2
  - **Randomness complexity**:
    $\left((2^{\ell} \cdot (n - \ell)) + m \cdot (2^{(n-\ell)} + 2^{\ell})\right)$-bits
  - Use *three-wise* independent PRG [*TS09, IKL+13*] to reduce number of true random values
  - Compute masks *on-the-fly*

Second-Order Lookup Table Compression

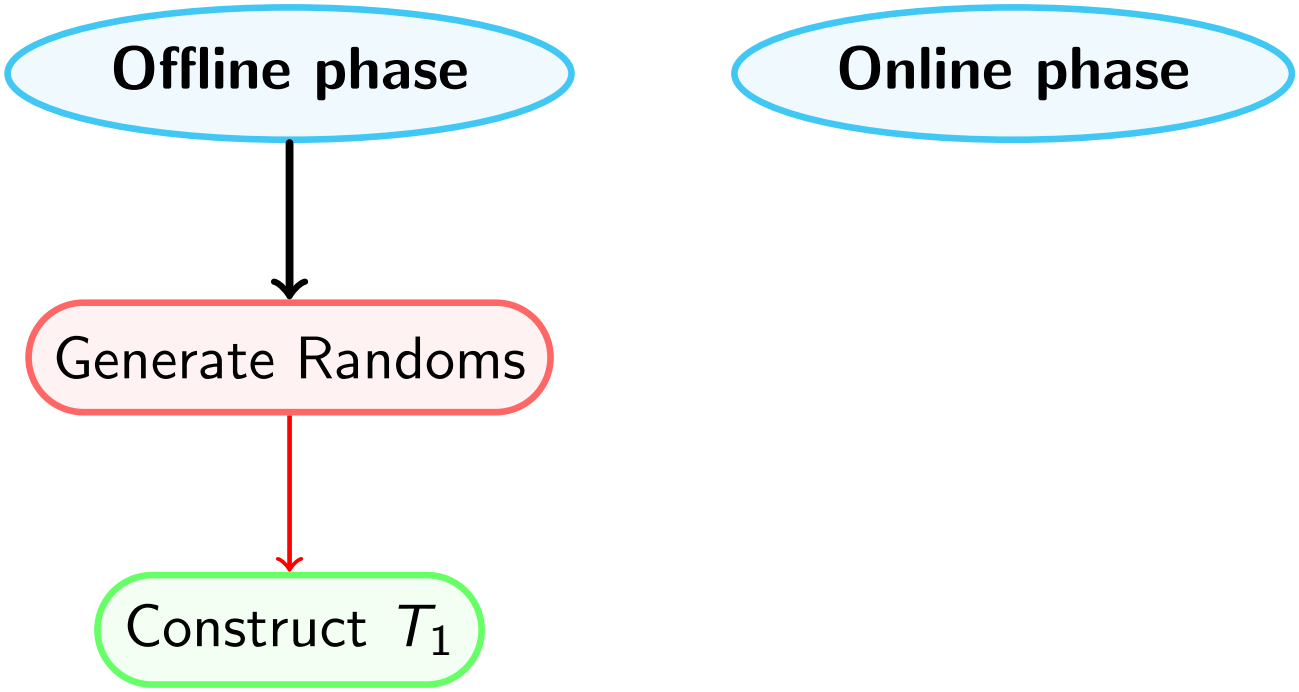
- Following are the highlights of our scheme
  - Randomise rows of tables T_1 and T_2
  - **Randomness complexity**:
    $\left((2^{\ell} \cdot (n-\ell)) + m \cdot (2^{(n-\ell)} + 2^{\ell})\right)$-bits
  - Use *three-wise* independent PRG [*TS09, IKL+13*] to reduce number of true random values
  - Compute masks *on-the-fly*
  - Make provision for *pre-processing*

High-Level Overview of Our Scheme

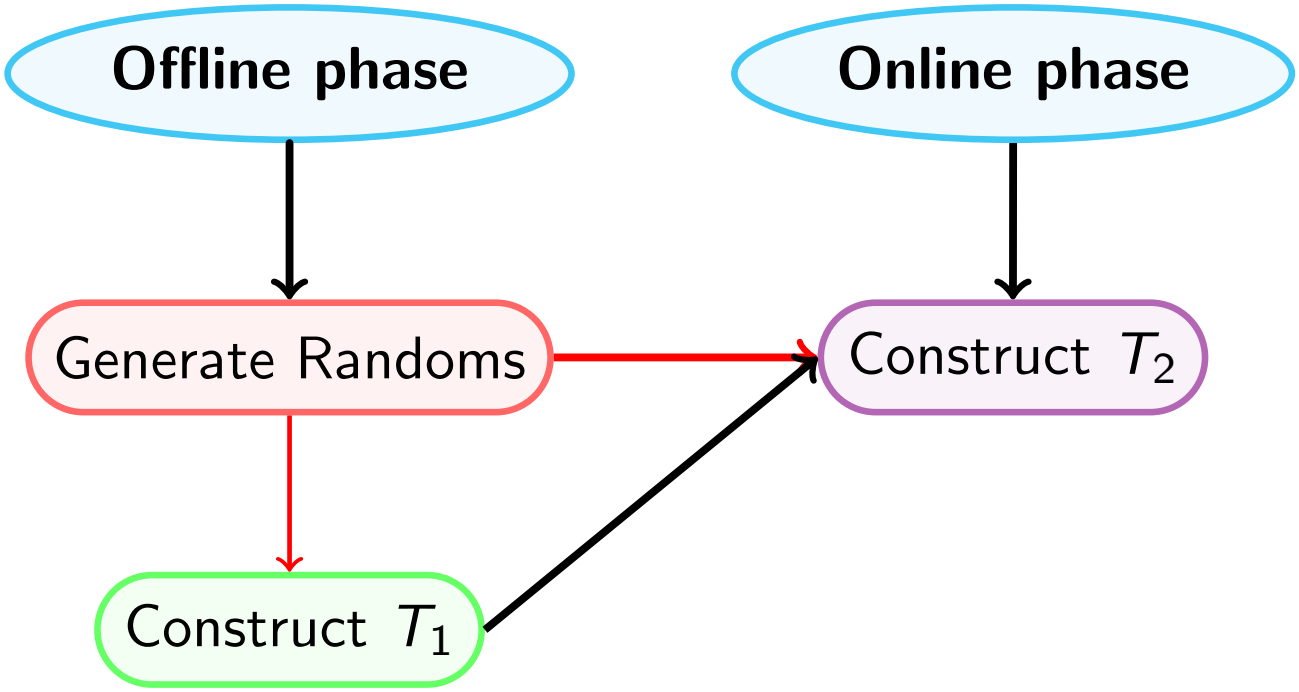
Offline phase

Online phase

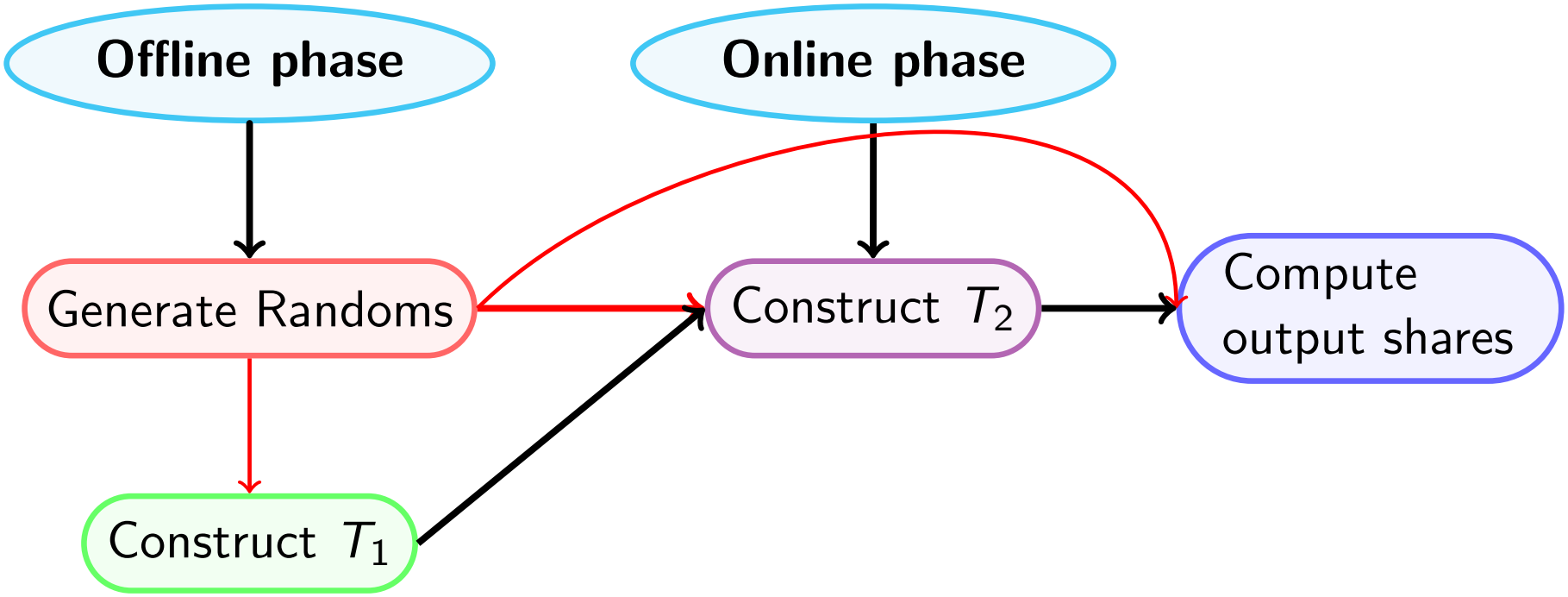
High-Level Overview of Our Scheme



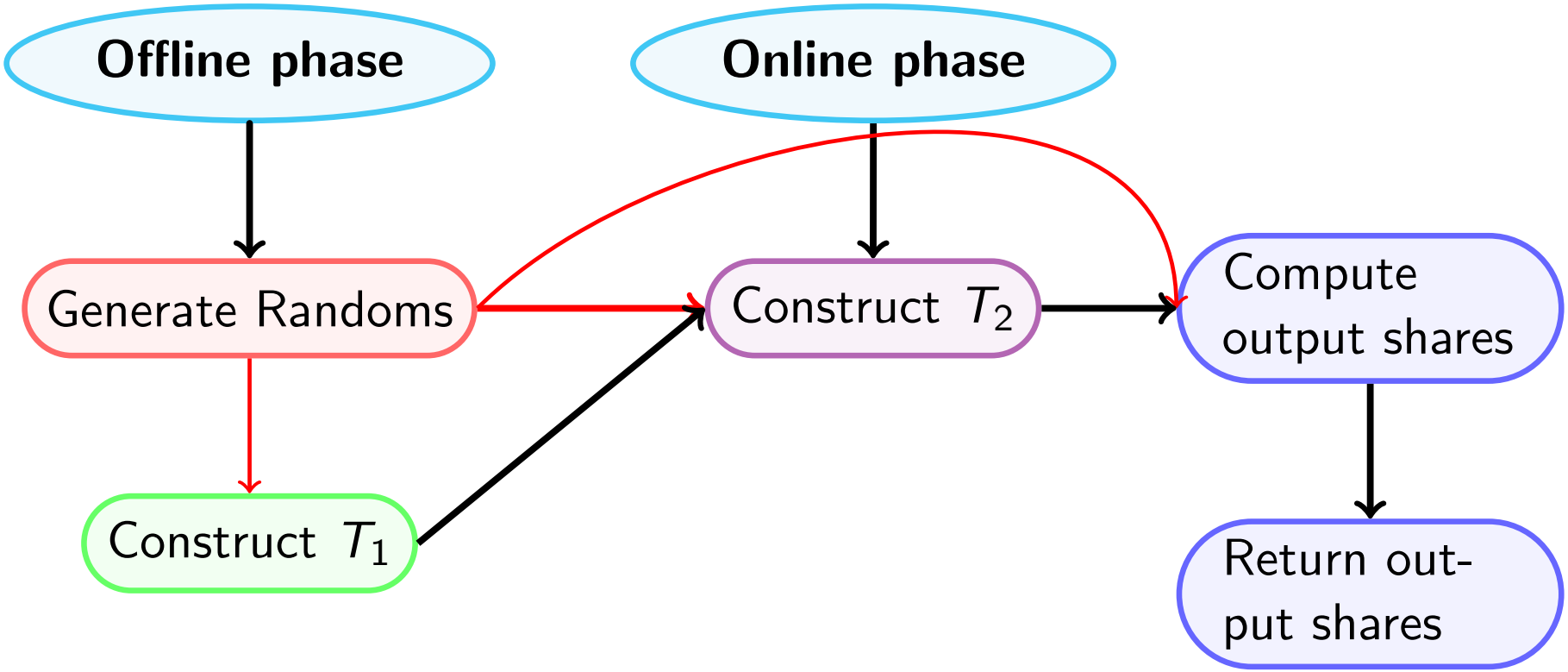
High-Level Overview of Our Scheme



High-Level Overview of Our Scheme



High-Level Overview of Our Scheme



Security of Our Scheme

- Security in probing leakage model under *composition* [*BBDFGSZ'16*]

Security of Our Scheme

- Security in probing leakage model under *composition* [*BBDFGSZ'16*]
- We prove that any pair of values can be *simulated* independent of secret

Security of Our Scheme

- Security in probing leakage model under *composition* [*BBDFGSZ'16*]
- We prove that any pair of values can be *simulated* independent of secret
- Observations:

Security of Our Scheme

- Security in probing leakage model under *composition* [*BBDFGSZ'16*]
- We prove that any pair of values can be *simulated* independent of secret
- Observations:
  - Is it possible to prove security of T_1 and T_2 separately?

Security of Our Scheme

- Security in probing leakage model under *composition* [*BBDFGSZ'16*]
- We prove that any pair of values can be *simulated* independent of secret
- Observations:
  - Is it possible to prove security of T_1 and T_2 separately?
  - Simulation of pairs using *shared* random values

Security of Our Scheme

- Security in probing leakage model under *composition* [*BBDFGSZ'16*]
- We prove that any pair of values can be *simulated* independent of secret
- Observations:
  - Is it possible to prove security of T_1 and T_2 separately?
  - Simulation of pairs using *shared* random values
  - Formal verification [*BBFG'19, Cor'18*]

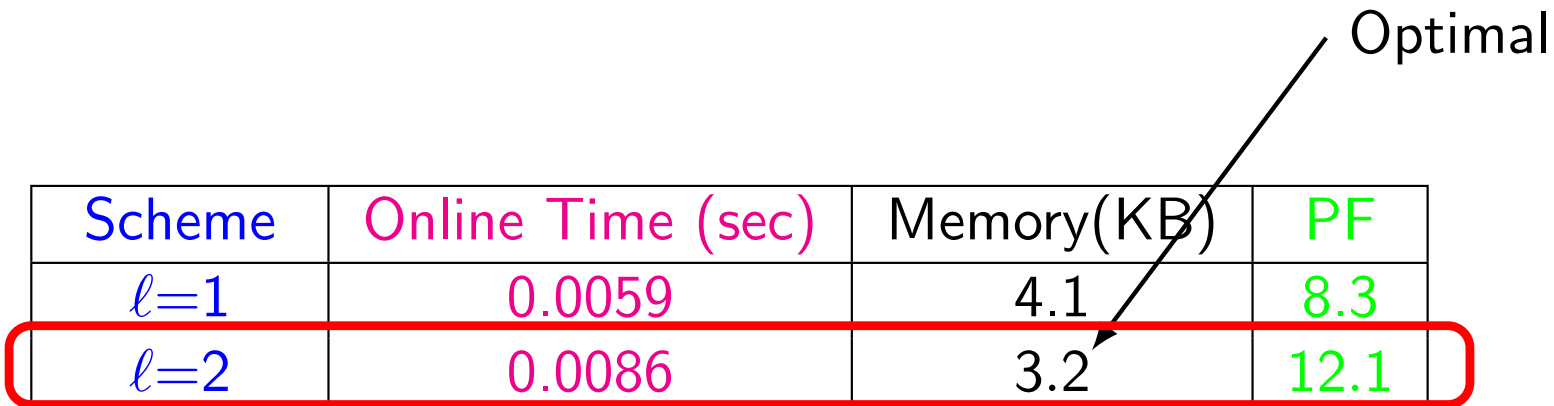
Implementation Results

Results for AES-128

Optimal

Scheme	Online Time (sec)	Memory(KB)	PF
ℓ=1	0.0015	22.5	11.8
ℓ=2	0.0023	12.7	18.2
ℓ=3	0.0057	8.1	44.9
[RP'10]	0.010638	0.03	83.6
bitslicing	0.008108	1.1	63.7
[RDP'08]	0.00721	0.3	56.6
RDP Preprocessing	0.000606	40.6	4.8
[CRZ'18]	0.017792	120.2	139.7

Results for PRESENT



Scheme	Online Time (sec)	Memory(KB)	PF
ℓ=1	0.0059	4.1	8.3
ℓ=2	0.0086	3.2	12.1
ℓ=3	0.0107	2.1	15
[CRV'15]	0.0089	0.04	12.6

Conclusion

- Using our scheme, AES S-box requires a memory of 59 bytes to store randomised table (reduced by a factor of 4)
- With pre-processing, 2-O secure AES-128 requires a memory of 8.1 KB only instead of 40 KB using [*RDP'08*]
- AES S-box requires 19-bytes of true randomness per call (optimal case)
- For generic S-box implementation, our scheme strikes a balance between memory and online time (taking advantage of pre-processing)

Scope for Future Work

- How feasible it is to extend the compression technique to higher-order schemes
- Optimisation of masking schemes to resource-constrained devices

https://eprint.iacr.org/2020/879.pdf

References

[*ISW'03*] Y. Ishai, A. Sahai, D. Wagner. *Private circuits: Securing hardware against probing attacks*. CRYPTO'03.
[*CJRR'99*] S. Chari, C.S. Jutla, J.R. Rao, P. Rohatgi. *Towards sound approaches to counteract PAA*. CRYPTO'99.
[*RP'10*] M. Rivain, E. Prouff. *Provably secure higher-order masking of AES*. CHES'10.
[*DDF'14*] A. Duc, S. Dziembowski, S. Faust. *Unifying Leakage Models: From Probing Attacks to Noisy Leakage*. EUROCRYPT'14.
[*CJRR'99*] S. Chari, C.S. Jutla, J.R. Rao, P. Rohatgi. *Towards sound approaches to counteract PAA*. CRYPTO'99.
[*PR'07*] E. Prouff, M. Rivain. *A generic method for secure Sbox implementation*. WISA'07.
[*RRST'02*] J.R. Rao, P. Rohatgi, H. Scherzer, S. Tinguely. *Partitioning attacks: Or how to rapidly clone some GSM cards*. IEEE S&P'02.
[*Vad'17*] P.K. Vadnala. *Time-memory trade-offs for side-channel resistant implementations of block ciphers*. CT-RSA'17.
[*TS'09*] Tamir Tassa and Jorge L. Villar. *On Proper Secrets, (t, k)-bases and linear codes*. Design Codes Crypto.
[*IKL+13*] Ishai *et al.*. *Robust pseudorandom generators*. ICALP'13.
[*BBDFGSZ'16*] G Barthe *et al.*. *Strong Non-Interference and Type-Directed Higher-Order Masking*. ACM CCS'16.
[*BBFG'19*] G Barthe, S Belaïd, PA Fouque and B Grégoire. *maskVerif: a formal tool for analyzing software and hardware masked implementations*. ESORICS'19.
[*Cor'18*] JS Coron. *Formal Verification of Side-Channel Countermeasures via Elementary Circuit Transformations*. ACNS'18.
[*CRZ'18*] JS Coron, Franck Rondepierre, Rina Zeitoun. *High Order Masking of Look-up Tables with Common Shares*. CHES'18.
[*CRV'15*] JS Coron, Arnab Roy, Srinivas Vivek. *Fast Evaluation of Polynomials over Binary Finite Fields and Application to Side-Channel Countermeasures.*. J. Cryptographic Engineering'15.
[*RDP'08*] M Rivain, E Dottax, E Prouff. *Block Ciphers Implementations Provably Secure Against Second Order Side Channel Analysis.*. FSE'08.

Images

Sources of images:

- Lock with broken shadow image (on S. 5): https://zdnet1.cbsistatic.com
- Side channel attack (on S. 6): https://www.togawa.cs.waseda.ac.jp
- Adversary probing (on S. 8): https://www.cryptoexperts.com
- Table (on S. 13): https://5.imimg.com
- Seesaw (on S.11): https://www.dreamstime.com
- Abacus (on S.13): https://www.gettyimages.com